# KAREL THE ROBOT

Equal Chamber Divisor

ABSTRACT

This study presents a method for dividing worlds into four equal chambers, accommodating special cases where four equal chambers are unattainable.

NOUREDDIN MOHAMMAD AHMAD LUTFI

Atypon Java and DevOps

# Outline

# Key Concept

According to Karel the Robot documentation "Karel is a very simple robot living in a very simple world. By giving Karel a set of commands, you can direct it to perform certain tasks within its world. The process of specifying those commands is called programming".

In this assignment we will utilize these commands to divide any given world into the biggest 4 equal champers or into (3, 2, 1) biggest equal chambers respectively if dividing is unattainable. Given that we need to optimize number of Karel moves and number of beepers used. For example, a divided 10 x 10 world is shown below.

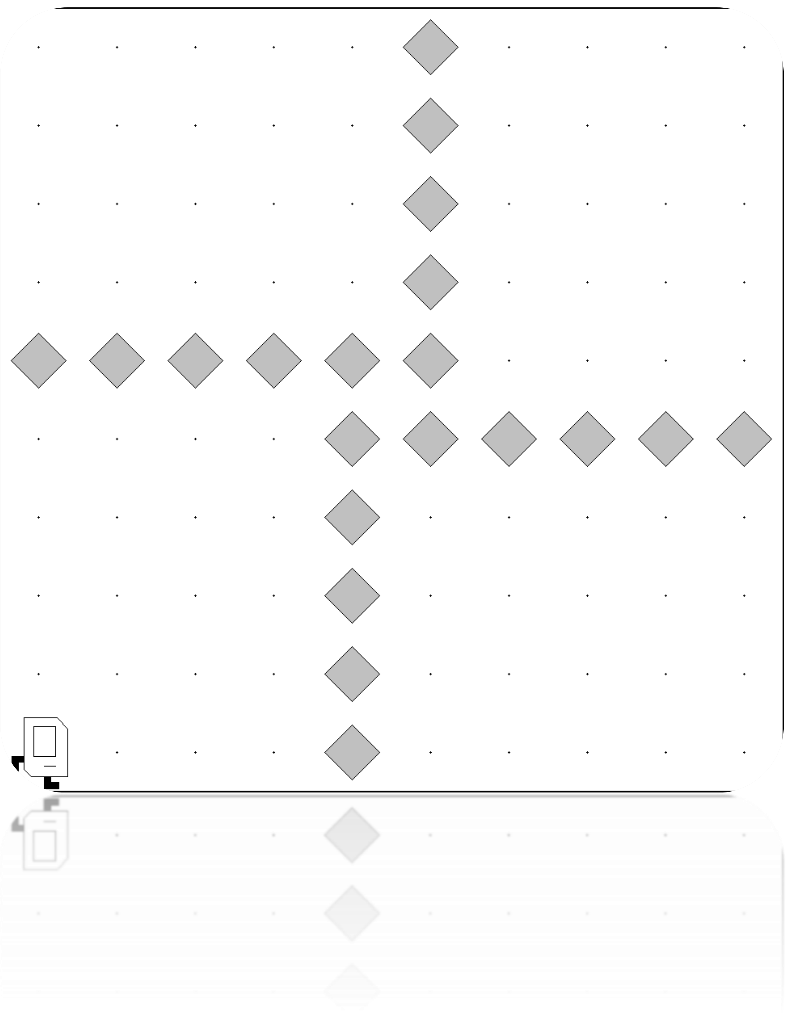


*Figure 1. 10 x 10 Divided World*

Fig 1 shows an example of dividing a world into 4 equal chambers; however, this will not be the case for all worlds so we will summarize the solution into three different cases.

## First Steps

First, we move Karel all the way to the right to determine the horizontal dimension. After we do so, we turn it left and move one step to determine if the vertical dimension is <= 2, we will call Karel at this position "Reference Position".

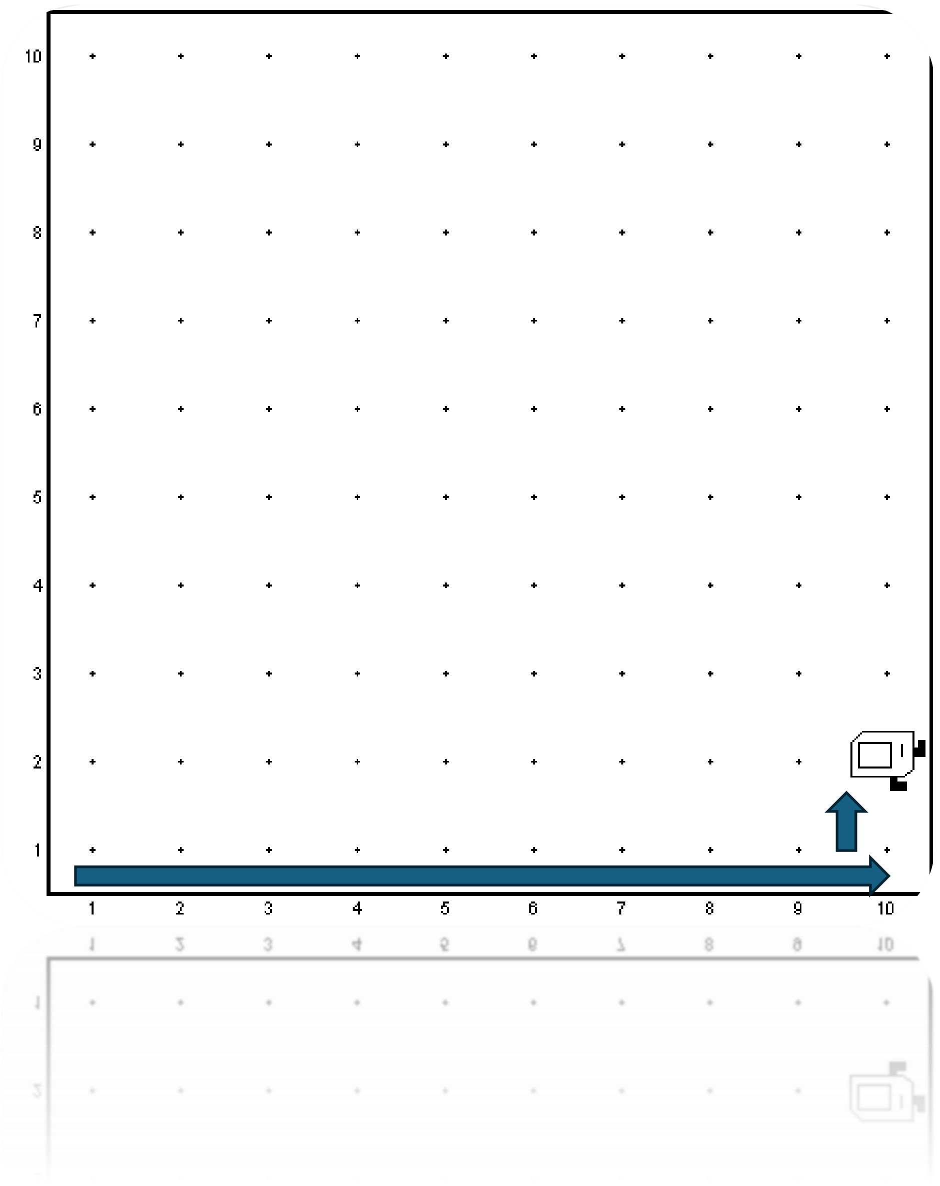


*Figure 2. Karel Base Algorithm First Steps*

Once Karel is here, we have two information, Horizontal Dimension & whether Vertical dimension is <= 2, which will lead us to one of the three cases we mentioned before. Also, Karel has a horizontal moves counter, vertical moves counter, and all moves counter. Moreover, Karel always to return to origin.

# World Cases

In this section, we will illustrate the three main cases that we will generalize our solution to:

## ❖ Case 1: Vertical Dimension = 2 or Vertical Dimension = 1 (1 or 2 rows only)

In this case we have two possible scenarios.

- Scenario I: Vertical Dimension = 2 and 1 < Horizontal Dimension <7 (2 rows, columns <7)

In Scenario I, which we will call "Horizontal Zigzag", we will use the "zigzag" dividing, which will maximum chambers to divide into (4,3,2), but we will have some padding in some cases. The following figures show examples of how we divide the world in the zigzag case. Assum Karel was at Reference Point.

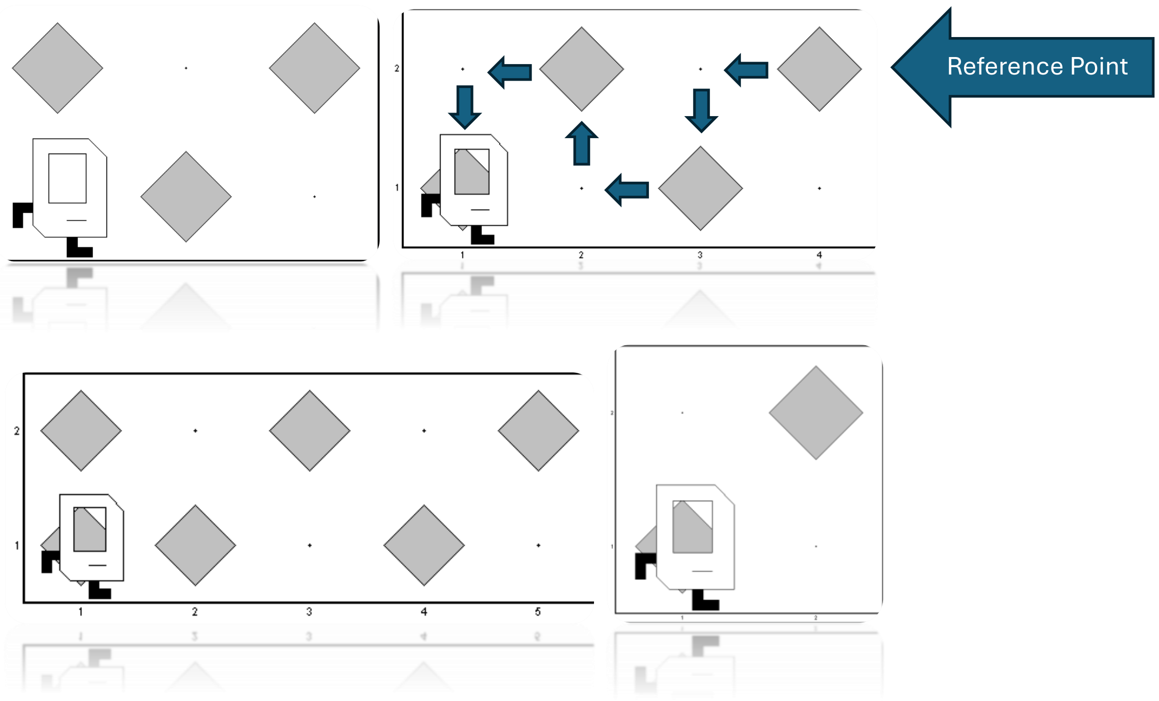


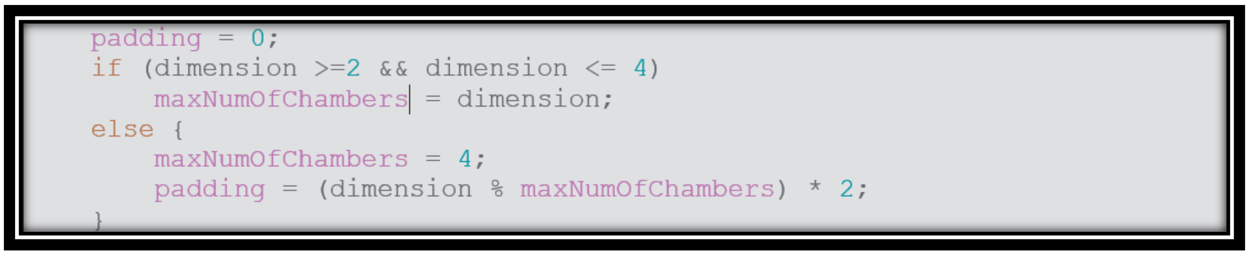
*Figure 3. Horizontal Zigzag Examples*

Now, the following table summarizes the possible instances above, which we will derive our approach from.

Table 1. World Division Parameters for First Dimension = 2

| Other Dimension | Maximum Number of Chambers | Number of Needed Walls | Padding Beepers | Champer Size |
|---|---|---|---|---|
| 2 | 2 | 2 | 0 | 1 |
| 3 | 3 | 3 | 0 | 1 |
| 4 | 4 | 4 | 0 | 1 |
| 5 | 4 | 4 | 2 | 1 |
| 6 | 4 | 4 | 4 | 1 |

Now we can calculate a variable called Padding, which represents the number of padding lines needed to be added, and Maximum Number of Chambers using the following sample code snippet, "dimension" represents "other dimension" in the table.



```
    padding = 0;
    if (dimension >=2 && dimension <= 4)
        maxNumOfChambers = dimension;
    else {
        maxNumOfChambers = 4;
        padding = (dimension % maxNumOfChambers) * 2;
    }
```

*Figure 4. World Division Parameters for Dimension = 2*

## Approach:

- Move in zigzag shape until reaching the maximum number of chambers value.
- Fill all remaining blocks with padding if needed.

- Scenario II: Otherwise (any scenario that applies to case 1 but not Scenario I)
  In this case we will use "Horizontal Dividing", we will calculate parameters using other formulas, then we simply apply them in code to get the following shapes.

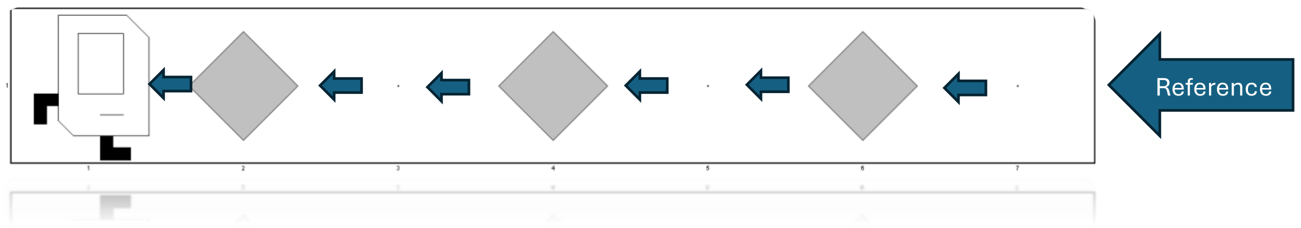


*Figure 5. 7x1 Divided World*

`

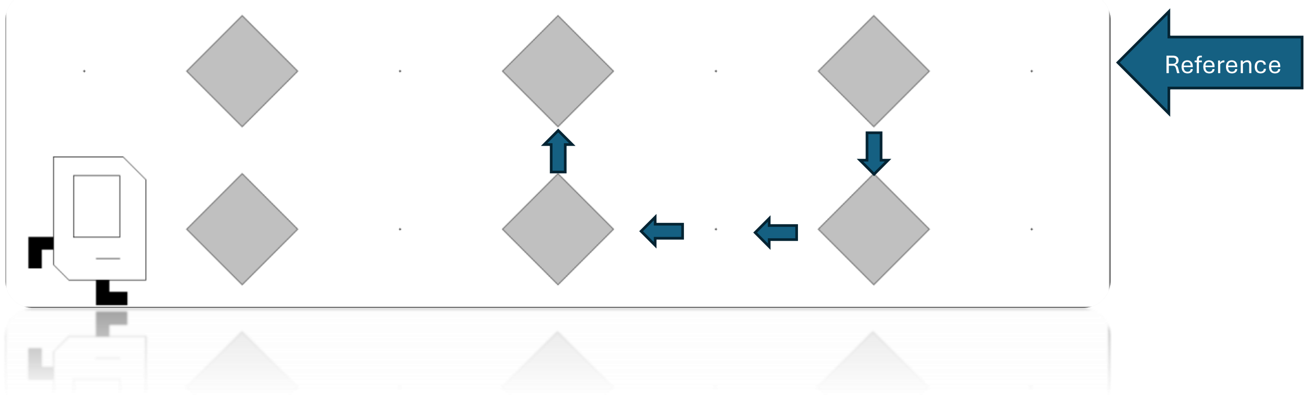


*Figure 6. 7x2 Divided World*

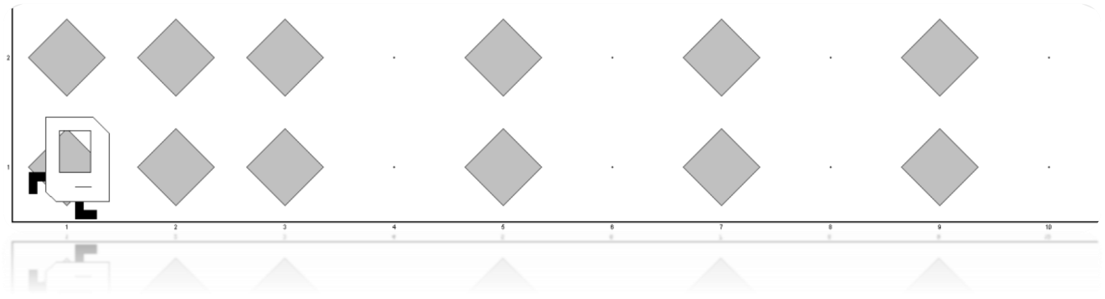


*Figure 7. 10x2 Divided World*

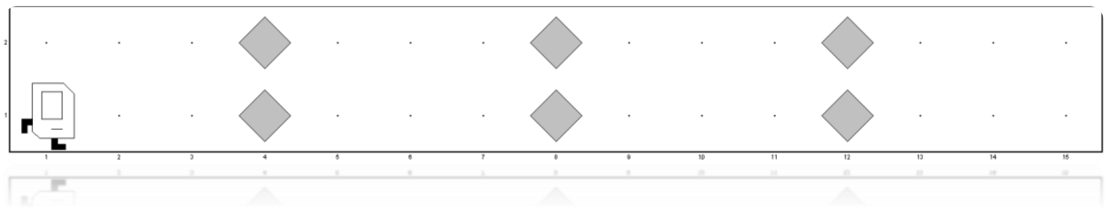


*Figure 8. 15x2 Divided World*

Here is the following table which summarizes the figures above which our approach will be derived from.

Table 2. Division Parameters for First Dimension = 2

| Other Dimension (OD) | Maximum Number of Chambers | Number of Needed Lines | Padding | Hop (Distance between beepers) |
|---|---|---|---|---|
| 7 | 4 | 3 | 0 | 1 |
| 8 | 4 | 3 | 1 | 1 |
| 9 | 4 | 3 | 2 | 1 |
| 10 | 4 | 3 | 3 | 1 |
| 11 | 4 | 3 | 0 | 2 |
| 12 | 4 | 3 | 1 | 2 |
| 13 | 4 | 3 | 2 | 2 |
| 14 | 4 | 3 | 3 | 2 |
| 15 | 4 | 3 | 0 | 3 |
| 16 | 4 | 3 | 1 | 3 |
| 17 | 4 | 3 | 2 | 3 |
| 21 | 4 | 3 | 2 | 4 |
| 26 | 4 | 3 | 3 | 5 |
| 30 | 4 | 3 | 3 | 6 |

From the table above and taking the dimension = 1 into consideration, we can conclude the following equations:

- Maximum Number of Chambers = $\begin{cases} 4; \text{ Other Dimension} >= 7 \\ \frac{OD+1}{2}; \text{ Other Dimension} < 7 \end{cases}$

- $\text{Number of Needed Lines} = \text{Maximum Number of Chambers} - 1$

- $\text{Padding} = (\text{OD} + 1)\ \%\ \text{Maximum Number of Chambers}$

- $\text{Hop} = \frac{OD - (Padding + Number\ of\ Needed\ Lines)}{(\text{Maximum Number of Chambers})}$

Important Note: I used the words Dimension and other Dimension

because this equation is valid for other cases that we will see later.

## Approach:

since Karel is at the reference point, Karel start dividing the horizontal dimension into lines according to the equation, turn Karel while moving, then fill with padding if needed.

Example:

i) Maximum Number of Chambers = 4

ii) Number of Needed Lines = 3

iii) Padding = 0; According to the formula above

iv) Hop = 1; According to the formula above

Karel is at Reference Position

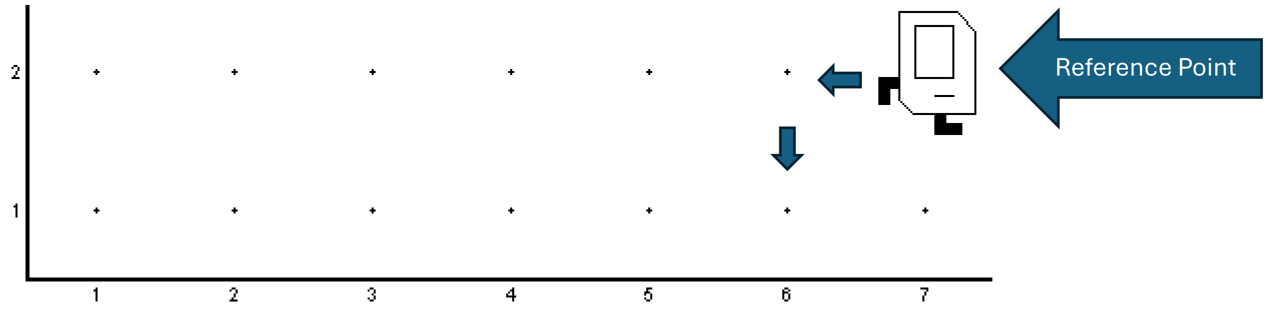


*Figure 9. Karel at Reference Position*

Karel takes a step and divides the first chamber.

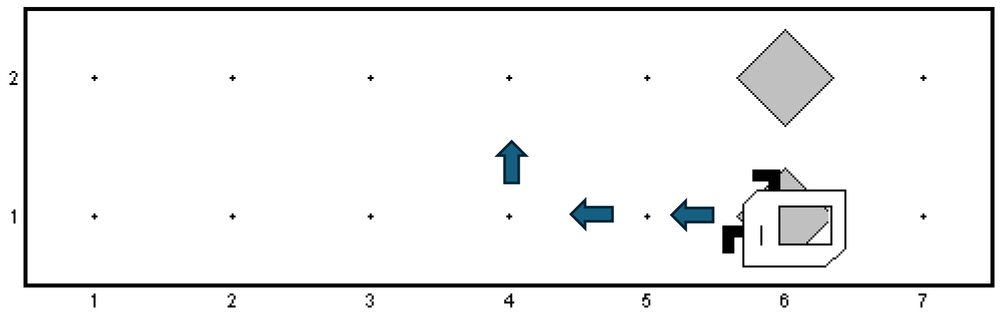


*Figure 10. Karel dividing first chamber*

Karel takes a step and divides the second chamber, and he moves on until the end of the world.

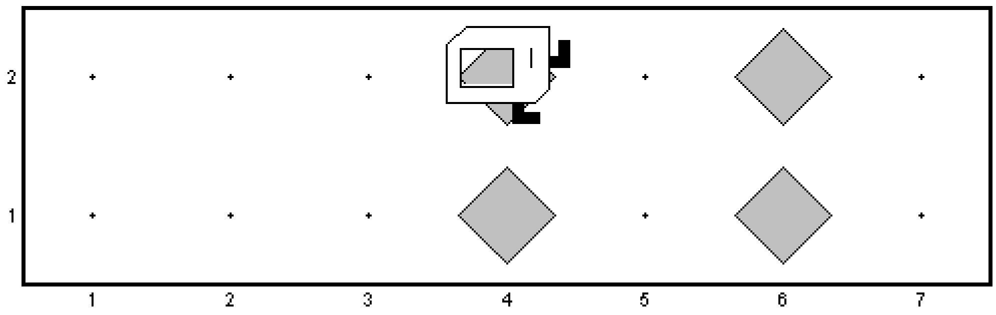


*Figure 11. Karel dividing second chamber*

Finally, Karel Divides last chamber and adds padding only if needed. For example, we didn't need any padding here. Finally, he gets back to origin.

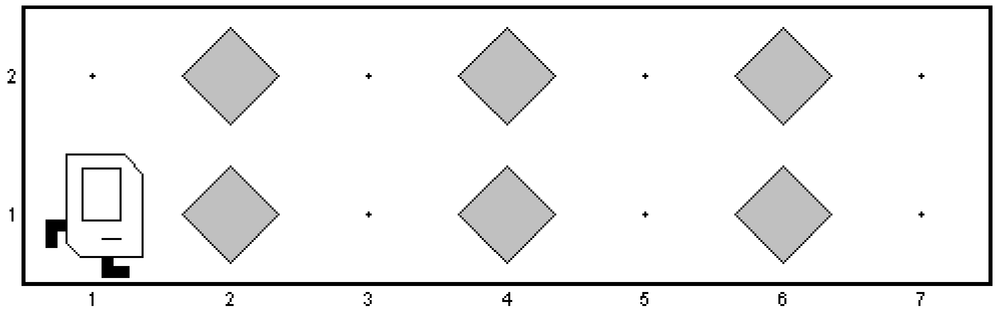


*Figure 12. Karel dividing second chamber*

❖ Case 2: Horizontal Dimension = 2 or Horizontal Dimension = 1 (1 or 2 columns only)

We move Karel to the top to determine Vertical Dimension, the figure on the left below illustrates the situation. The other two figures represent the same 2 scenarios and same approach as before, but the only difference that the world is flipped.

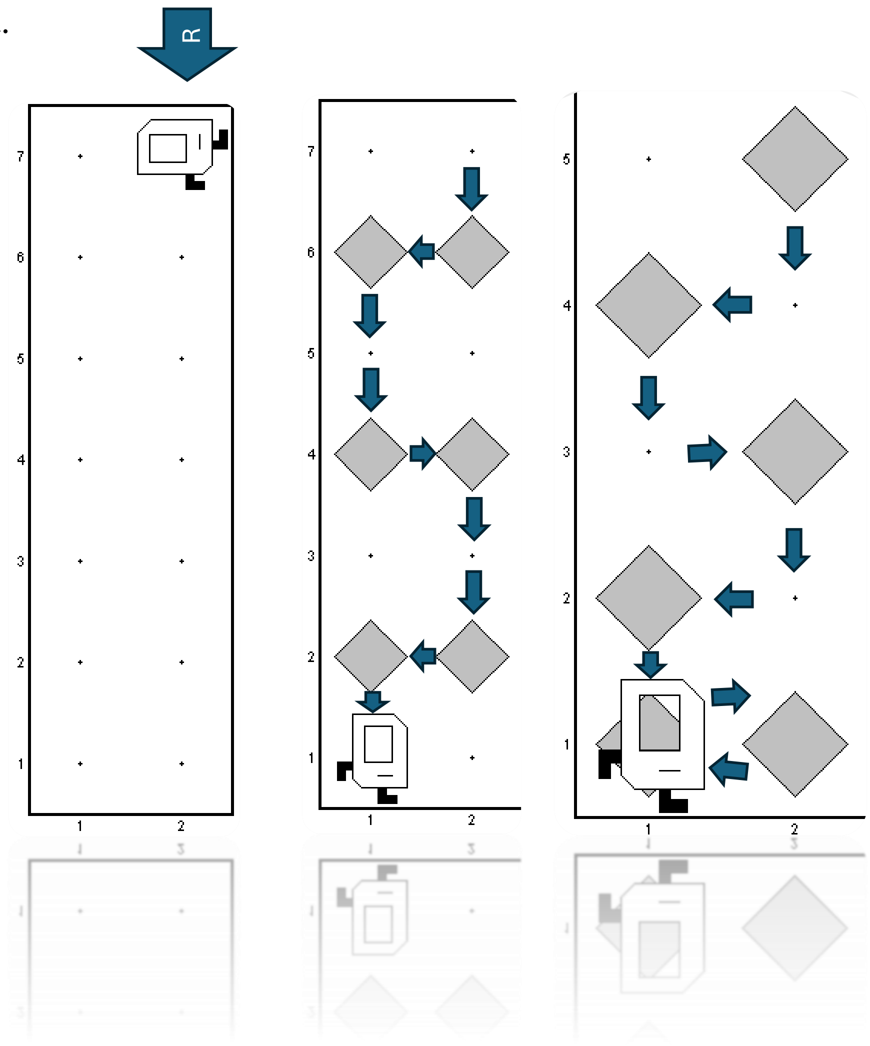


*Figure 13. Karel dividing Vertical Worlds*

Hence, we can solve using the same logic (same parameters calculation method), but we must use other functions to turn Karel while moving whether in zigzag or horizontal dividing, so we end up with "Vertical Zigzag" and "Vertical Dividing", many optimizations take place here since we use the same calculations and only change the moving direction and turning direction.

### ❖ Case 3: Otherwise

First, Karel is at the "Reference point" that we mentioned before.

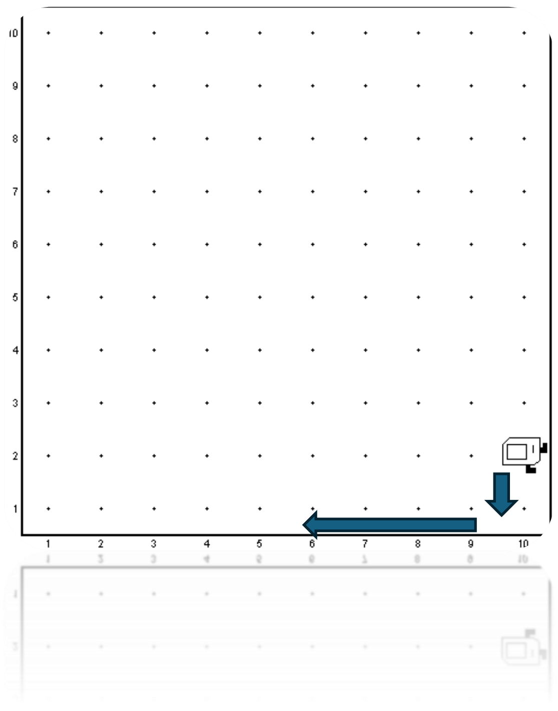


*Figure 14. Karel at Reference Point*

Then, Karel will turn around and move one step then turn right and move until he reaches the middle of the world.

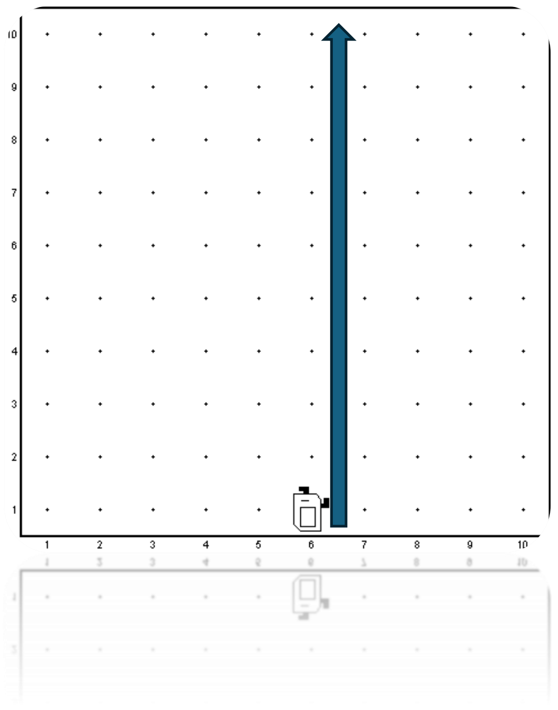


*Figure 15. Karel at the middle of the world*

Finally, Karel turn right, and finally move to the top while putting beepers so we end up in the top middle of the world. We will call this case "Divide big worlds", Horizontal dimension >= 3 & Vertical dimension>= 3, and Karel is at "Reference Point 2".

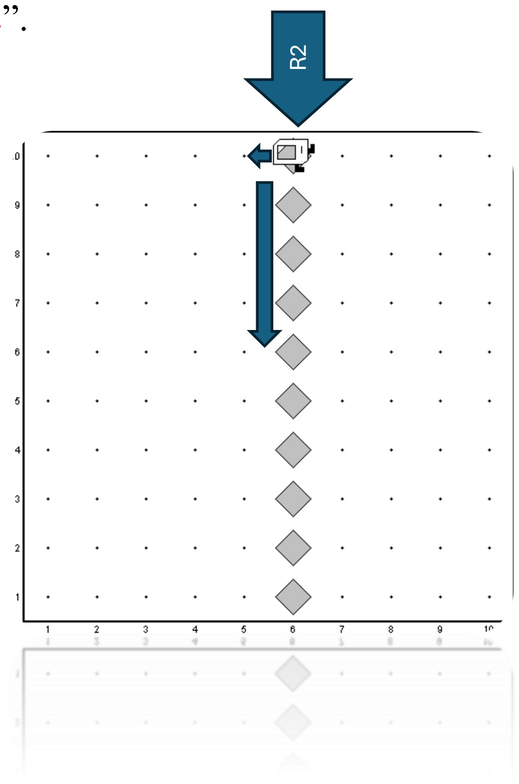


*Figure 16. Karel Draws the first Line*

After Karel reaches Reference Point 2, he will choose one of the following scenarios:

- Scenario I: Horizontal Dimension == Vertical Dimension & they are even Karel will draw a fan-like shape by following the steps below.

  First, Karel will move one step left then move down for a calculated number of steps (hop) without any beepers.

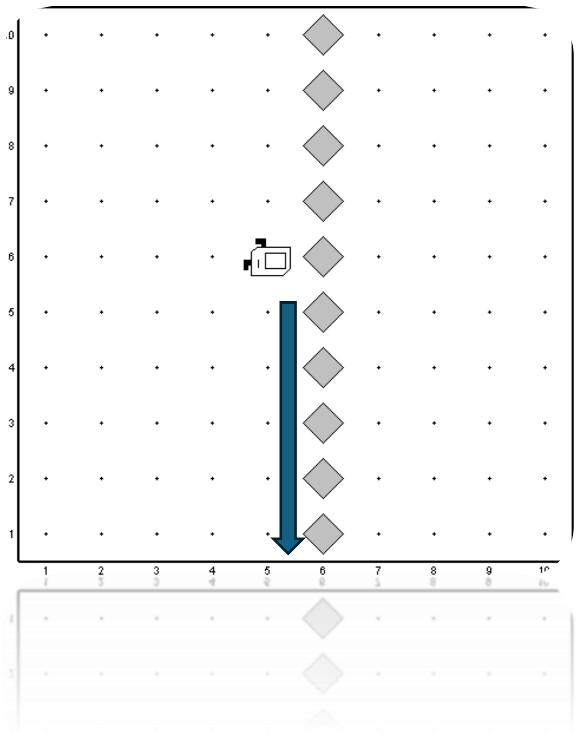


Figure 16. Karel Drawing Fan-shape Step 1

Then, he will keep moving forward while putting beepers.

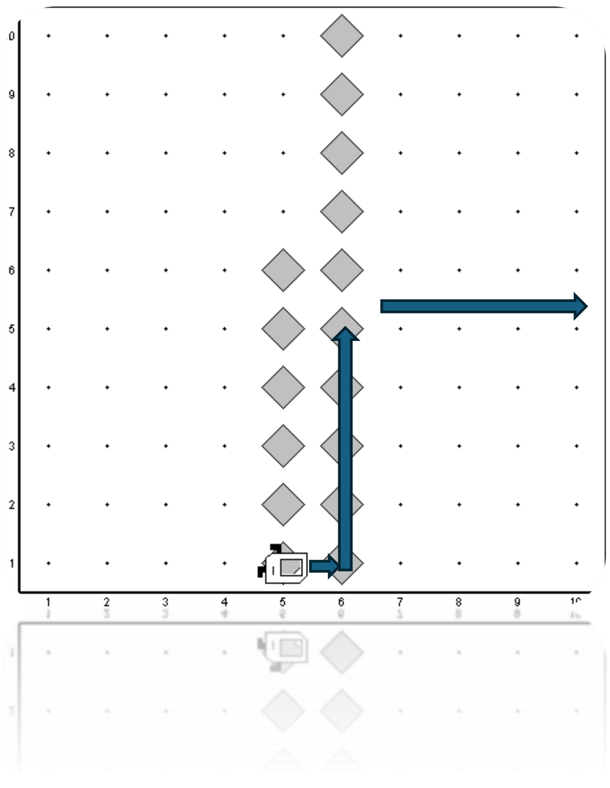


Figure 17. Karel Drawing Fan-shape Step 2

After that, Karel turns and removes the excess beepers, then draws the right line, even this is a drawback, but it happens only in this case given that in many other cases it will help us optimize number of movements.

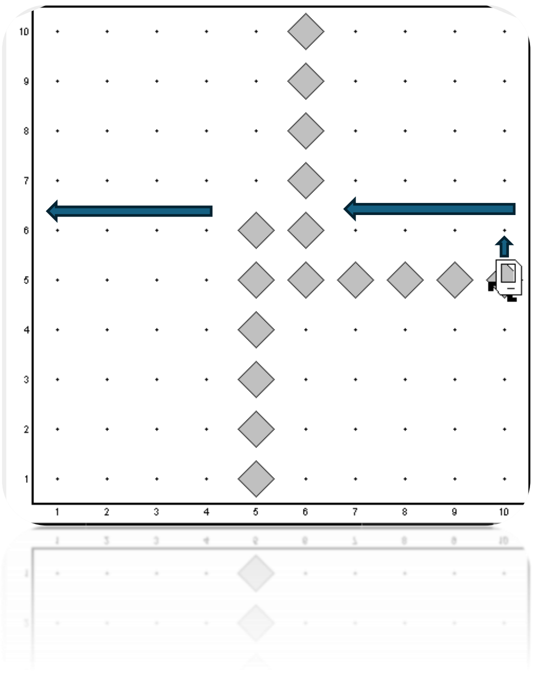


*Figure 18. Karel Drawing Fan-shape Step 3*

Finally, Karel will turn and draw the last line, so we end up with the following fan-like shape.

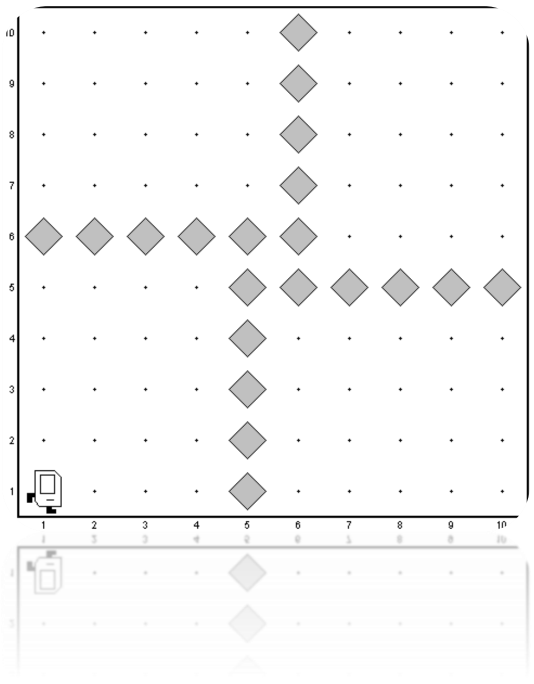


*Figure 19. Karel Drawing Fan-shape Final Step*

The following table shows the parameter (hop), which represents the number of skipped locations (not filled with beepers) from each line.

Table 3 . Map Division Parameters for equal even maps

| Dimension | Hop |
|---|---|
| 4x4 | 1 |
| 6x6 | 2 |
| 8x8 | 3 |
| 10x10 | 4 |
| 12x12 | 5 |
| 14x14 | 6 |
| 16x16 | 7 |

$$\text{Hop} = 1 + \frac{Dimension - 4}{2}$$

- Scenario II: Horizontal Dimension != Vertical Dimension || (Horizontal Dimension == Vertical Dimension && Dimension not even)
  This last scenario represents when Karel is at Second Reference Point, but the previous condition didn't apply, so Karel will behave according to the following condition.
  - Draw double lines for any even Dimension.
  - Draw a single line for odd Dimension.

  Details could be omitted, because they are very similar.

  Note: After we draw the vertical lines, we always draw the right lines before the left lines for optimization purposes, since at the end Karl will return to (1,1).

Example:

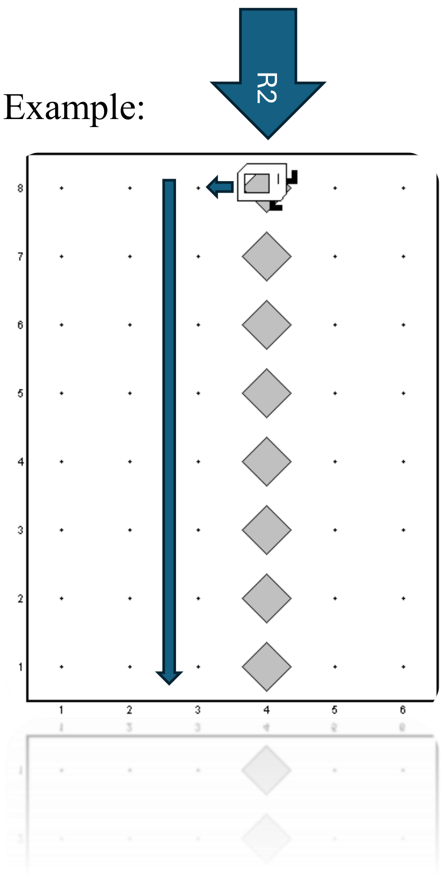


Figure 20. Karel at Reference point 2

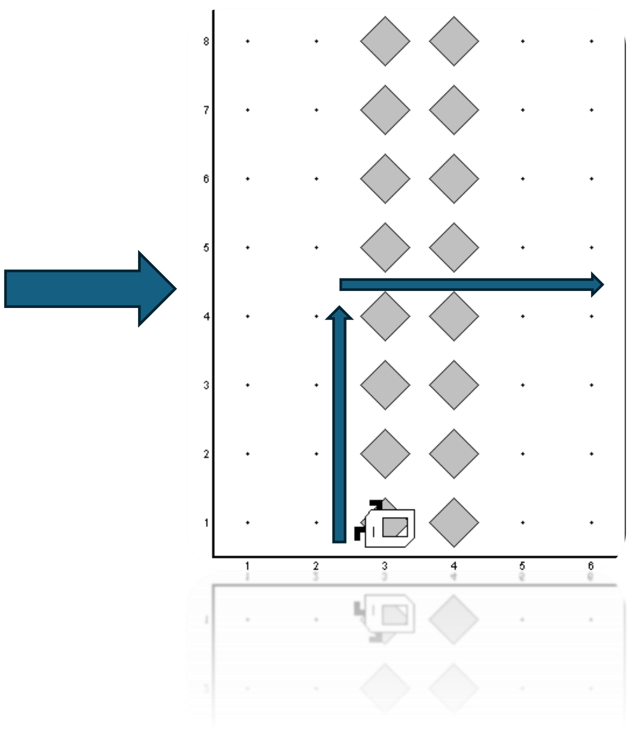


Figure 21. Karel Draws a double line

Karel turns and moves to the middle then draws half a line on the right as shows figure 22, then turns again to draw a whole line as shows figure 23.

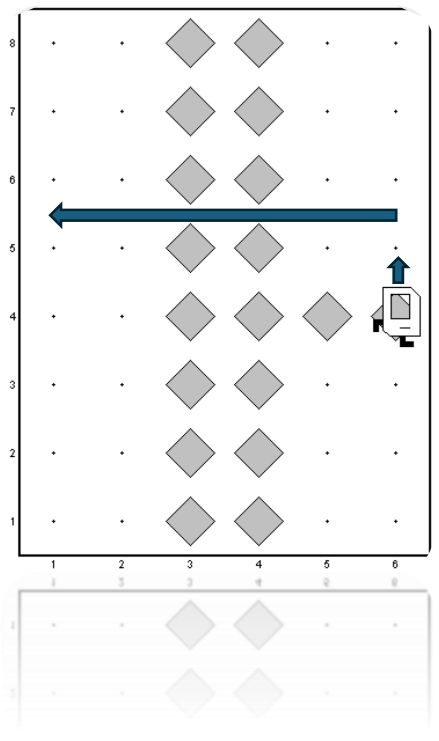


Figure 22. Karel draws half a line on the right

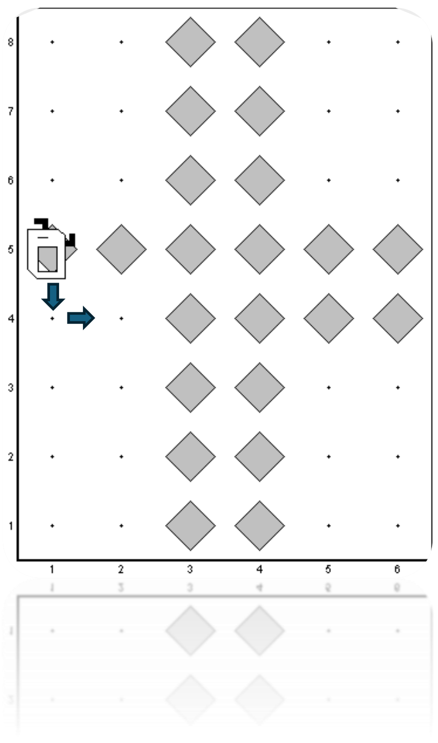


Figure 23. Karel Draws line

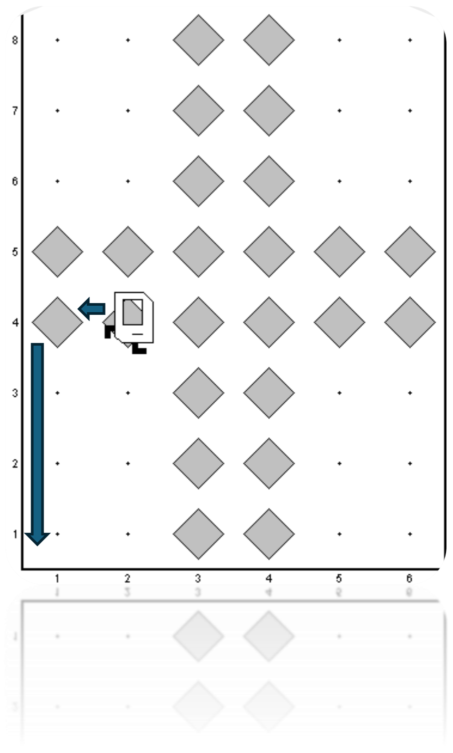


Figure 24. Karel turns and completes the shape

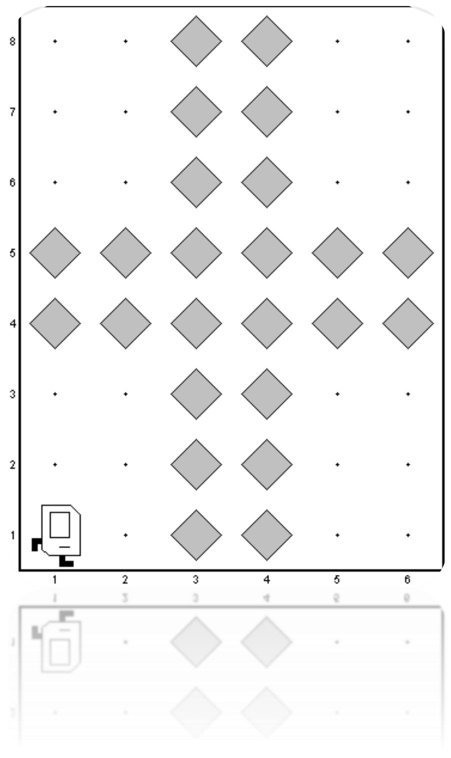


Figure 25. Karel returns to origin

The only difference is that when we have odd dimension, it will draw a single line not double line to divide that dimension.

More Examples:

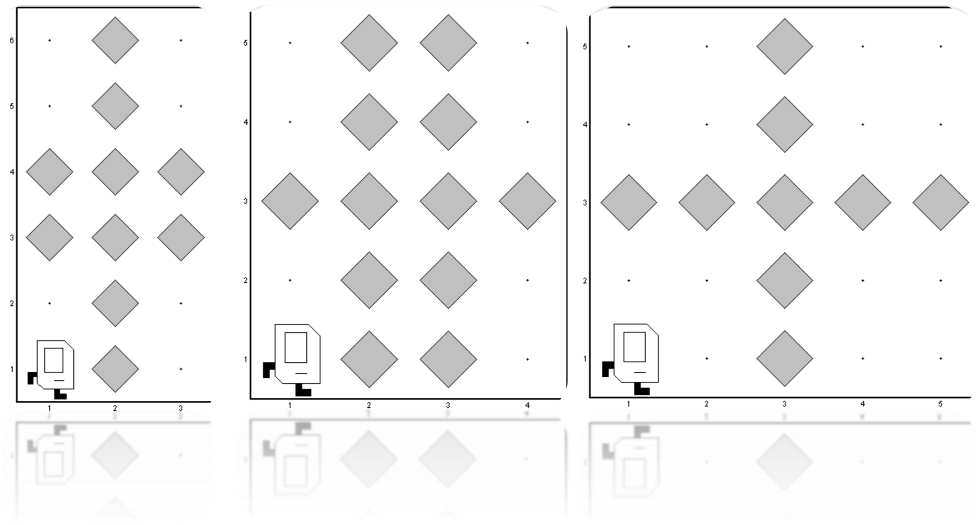


Figure 26. More Example on Scenario 2

These cases show why I choose to start dividing from middle before knowing the exact vertical dimension, because it will optimize number of movements instead of moving first to calculate number of movements.

Although, this will give a drawback in the fan-like shape, but it will optimize all other world cases. In other words, all maps are optimized but not when horizontal and vertical dimensions are even and equal, which is a single case, so it is worth it.

Finally, in all cases Karel turns left and moves all the way to the left, then he faces south and moves all the way down to return to origin.

## Debatable Case Analysis

One possible way of optimizing the chamber size is when horizontal and vertical dimensions are even and not equal.

For example, let's take the 6x8 world. Instead of dividing like the figure on the left,  it could be thought of like the one in the middle to achieve more chamber space and less beepers, or even more optimized like the one on the extreme right, but it might be hard to achieve, or hard to implement.

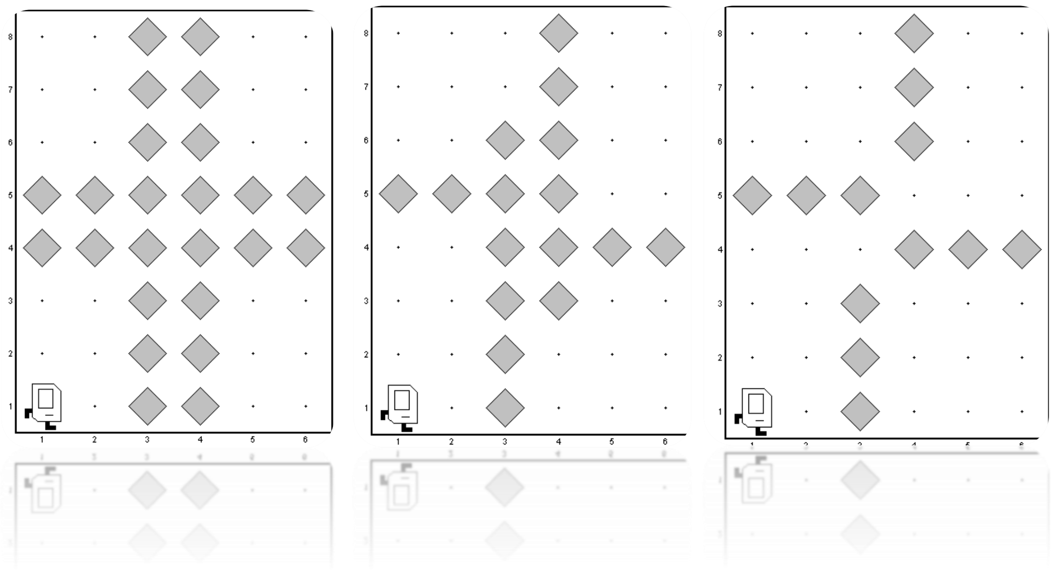


*Figure 27. Controversial Case Potential Solutions*

However, dividing this way might have several drawbacks:

- Not easy to implement since it is hard to find a general formula.
- Could make the code harder to read, adding more cases and more lines.
- The previous case is already simple and generalized (not much corner cases).
- The shapes contain 2 Irregular polygon and 2 rectangles, which could be controversial.

There could be a mathematical model that can represent this scenario, in such case it would be more convenient to use it to optimize the solution.